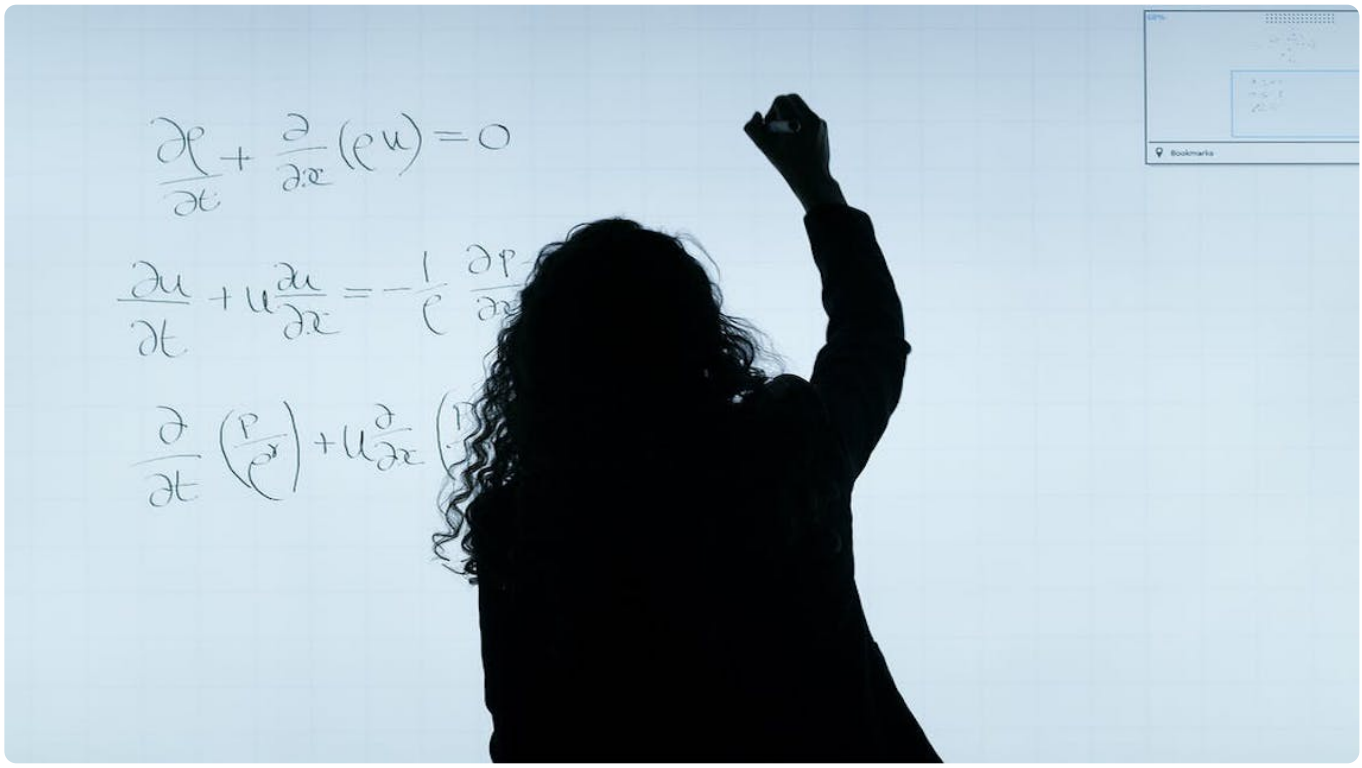
下载APP

12 | 文本处理函数：三招解决数据对齐问题

2021-03-09 尹会生

Python自动化办公实战课 进入课程 >



讲述：尹会生
时长 15:17 大小 14.00M

你好，我是尹会生。

当你在工作中利用 Excel 向同事展示业务数据时，肯定遇到过数据无法对齐的问题。

比如在展示日销售额报表数据时，如果数字、日期、文字这些常见的元素没有对齐，你不仅会因为格式显示混乱而难以分析数据变化的趋势，而且也会因为报表格式不够工整，给领导留下一个技术能力不扎实的不良印象。

而且，如果一个个手动来调整格式，不仅麻烦，而且还很容易出错。那么今天这节课，就教你怎样用 Python 的文本处理函数，利用 format()、split()、join() 和 strip() 四个函数来对 Excel 中的数字、日期和文字进行长度调整和移除多余空格等处理，解决数据对齐问题。

数据对齐的思路

为了更好地掌握 Python 的文本处理函数，我先带你了解 Excel 中数据的默认对齐方式，以及需要使用 Python 中的哪些文本函数。

Excel 中的数据是有默认对齐方式的，对齐的方式一共有三种，分别是右对齐、左对齐和居中对齐。数据类型不同，默认对齐方式也不同。

数值型数据会自动靠右对齐，比如日期、时间、数字。

文本型数据会自动靠左对齐，比如汉字、字母、英文、引号开头的数字。

当你把数据粘贴到 Excel 中时，如果是同类型数据没有实现对齐，你就要考虑在粘贴到 Excel 之前对数据进行处理。

具体怎么处理呢？我们可以根据 Excel 的默认对齐方式，以及导入到 Excel 中的内容，来分情况考虑。

在右对齐的类型中，数字、日期、时间是我们接触最多的数值类型，而数字和日期、时间的处理方式又有差别，所以我们要分开来学习和掌握它们。

在左对齐的类型中，文字、字母、英文和引号开头的数字，它们的对齐方式相同，都是去掉前方的空格。所以在你掌握文字前后的空格处理之后，就可以举一反三地对其他三种数据类型实现左侧对齐。

还有最后一种对齐方式是居中对齐。居中对齐的场景只有一种，那就是作为主题使用。这种对齐方式可以直接通过手动操作来完成。

因此，我通过对数字、日期、文字三种类型的处理，可以让你应对大多数数据对齐的场景了。总结来说，那就是：

采用 format() 函数，实现数字对齐；

使用 split() 和 join() 函数，实现日期右侧对齐；

使用 strip() 函数，实现文本型数据的左侧对齐。

接下来我就详细讲解一下这三种解决方法。

数据对齐的办法

第一招 使用 format() 函数，实现数字对齐

数字没有对齐，主要是因为数字位数不同。所以我们可以通过补全数字位数的方法，来对齐数字。

我们来看一个例子。在这个例子中，我提供了 5 个浮点数，它们的内容和格式如下：

复制代码

```
12.34
123.456
1.2345
123456.78
123.4
```

在这组浮点数中，它们各自的小数位数也不同。如果直接粘贴到 Excel，会因为小数位数不同而无法直观地比较它们的大小。

在这种情况下，你可以使用 Python 的内置 format() 函数，把它们都保留至小数点后四位，这样既能保证每个浮点数的精度，又能按右侧对齐小数点。

那我对这 5 个浮点统一使用 format() 函数进行处理，它的代码是：

复制代码

```
string1 = 123.45
print("{:.4f}".format(string1))
# 123.4500
```

这段代码把“123.45”使用 format() 函数进行格式处理后，就会保留小数点后四位，输出“123.4500”。所以如果把浮点数用 format() 函数进行处理，再粘贴到 Excel 中，就可以实现自动右侧对齐了。

那我们接下来就学习一下 format() 函数的**调用方式和格式，看**format() 函数是怎么**来补齐数字小数的位数，并且调整整数的位置的。**

format() 函数是 Python 的 2.6 版本新增的内置函数，它增强了字符串的格式处理功能。你应该想到了，format() 函数是内置函数，所以它和我们之前学习过的 print()、int()、str() 函数一样，可以在 Python 中直接调用，不用预先定义和 import 导入。

使用 format() 函数对数字格式调整时，调用 format() 的 Python 语句是“{: 数字的格式}.format(要转换格式的数字)”，在这条语句里，“:”后面的**数字的格式包含**三个部分。

第一部分表示符号和空格，可以使用 +、-、(空格)。

“+”表示在正数前显示 +；

“-”表示负数前显示 -；

（空格）表示在正数前加空格。

第二部分表示宽度，可以使用. < > ^，“.”来保留小数点后的位数。

.4 表示保留小数点后四位；

^, <, > 分别是居中、左对齐、右对齐，后面也需要使用数字指定宽度，如果不满足宽度自动补充空格。

第三部分表示类型，常见的有“b、d、f”，分别表示二进制、十进制和浮点数。比如我可以使用如下代码，来表示整数 100 占用 6 个字符的宽度，并靠右侧对齐。

复制代码

```
print("{:6>d}".format(100))
#执行结果
(空格)(空格)(空格)100
```

可以看到，format() 函数有着非常丰富的参数，除了可以进行数字对齐外，还可以调整字符串格式，所以我也经常用 format() 函数在工作中进行格式化字符串操作。不过这些参数

和数字对齐的关系不大，我在这节课就不把它们一一列出来了，如果你想详细了解，可以参考官方文档。

接下来我们来看一下如果是日期类型，怎么使用字符串函数实现它的右侧对齐。

第二招 使用字符串函数，实现日期右侧对齐

日期对齐和数字对齐相比，最主要的区别就是它的年、月、日三个部分都可能出现不等长的情况。我给你举个例子，你一看就明白了。

复制代码

```
2021-3-18
21-3-18
2021-12-21
2021-3-18
21-3-1
```

在这 5 个日期中，由于年、月、日的位数不同，导致粘贴到 Excel 后没法实现右侧对齐。因此我们实现日期对齐的前提，就是要补齐日期。而补齐日期的过程，我们需要按照**先拆分日期、再调整格式、最后合并日期**的顺序。

我先把它的代码写出来，供你参考，然后再给你讲解补齐日期的过程。

复制代码

```
date_demo = [
    "2021-03-18",
    "21-3-18",
    "2021-12-21",
    "2021-3-8",
    "21-3-1",
]

for  dd in date_demo:
    # 拆分日期
    year, month, day = dd.split('-')
    # 调整格式
    if len(year)== 2 :
        new_year = 2021
    else:
        new_year = year

    month = "{:>02d}".format(int(month))
```

```
    day = "{:>02d}".format(int(day))
    # 合并日期
    new_date = [str(new_year), str(month), str(day)]
    new_date = "-".join(new_date)

    print(new_date)

# 执行结果
2021-03-18
2021-03-18
2021-12-21
2021-03-08
2021-03-01
```

通过这段代码的执行结果，可以看到我把长度不同的 5 个日期，按照年、月、日三个部分别进行了长度处理，并使用“-”作为日期分隔符，从而实现日期的等长。

接下来，我按照程序的执行顺序，为你详细讲解一下我是怎么对日期进行**拆分、调整和合并**的。

为什么要先进行拆分这一操作呢？因为在我们已知的函数中，没法对年、月、日分别进行调整。如果整体进行调整，比如调整年，很容易影响到月和日的数字，我把它称作**调整的副作用**。所以我就需要通过 split() 函数先对完整日期进行拆分。

split() 函数是我们这节课要重点掌握的一个函数，我来为你讲解一下它的主要用法。

按照学习其他函数的经验，你需要掌握 split() 函数的功能、参数和返回值。实际上对于所有函数的学习，我们都需要先关注函数功能、参数个数和类型、返回值个数和类型三个部分，之后再去学习参数函数里的业务逻辑功能。这其实也是掌握函数的通用做法。

split() 函数是对字符串进行拆分的函数，也是根据字符串中指定的分隔符，它的参数“-”就是拆分字符串的分隔符，它的返回值是拆分以后字符串形成的列表。

我在返回值中使用了连续赋值的功能，那么就可以通过以下代码来实现：

复制代码

```
year, month, day = dd.split('-')
```

这种方式，让变量 year、month、day 依次得到了列表中的三个元素。

由于在代码的后续运行过程中，我需要对每个变量进行单独处理，而使用列表还需要通过下标访问每个列表。那么这里我就用三个变量代替列表来作为 split() 函数的返回值。不过我用变量代替列表还有一个原因，那就是在代码的阅读上，使用变量方式更易于理解。

接下来是调整内容阶段，我依然使用 format() 函数对日期的每个部分进行长度上的调整。你可以通过代码来学习，如何使用 format 进行日期的每个部分调整的。

最后，我把调整好的年、月、日，使用 join() 函数进行合并。为了和原始字符串保持一致，我继续使用“-”作为连接符号，对函数的参数 new_date 列表进行连接。而且，join() 函数返回的类型是字符串，刚好和处理之前的类型保持一致，如果你的代码里包含了对日期进行操作的功能，调整长度后这些代码就可以保持不变。

此外，这里还有两点需要你注意：

1. 工作中经常会遇到一个字符串中多处需要修改的内容，例如刚才举的日期例子，你可以把 split() 和 join() 函数配合使用，拆分成多个变量，对变量进行处理后，再进行合并，减少处理的复杂度。
2. 在使用 split() 前，你还可以通过正则表达式来处理不规范的分隔符，让字符串每个字段都能按照相同的分隔符进行拆分。

第三招 使用字符串函数，实现文本型数据的对齐

Excel 中的文字默认是左侧对齐，那文字类型为什么会出现没法对齐的情况呢？主要是因为文字在开头或结尾包含了一个或多个空格。

对于这些空格，可以使用字符串自带的 strip() 函数来去掉。我把删除文字前后空格的代码写在下方，供你参考：

复制代码

```
string = "    广东省广州市    "
newstring = string.strip()
print(f"|{newstring}|")

# 输出结果
|广东省广州市|
```

在这段代码中，我通过内置的 strip() 函数，去掉了字符串前后的空格，并通过 f-string 调整了字符串的格式。

strip() 函数是我们这节课新学习的函数，它的主要功能就是自动移除字符串开头和结尾指定的字符，如果没有为 strip() 函数指定参数，默认会移除字符串开头和结尾连续的多个空格。

另一个新学习的函数是 f-string，我在代码的第三行使用了它，用于调整 newstring 字符串的输出结果。f-string 是在 Python3.6 版本引入的一种新的字符串，它的写法是在字符串前增加 f 关键字，增加后我们就能在一般的字符串中使用“{}”关键字，“{}”中允许使用变量、运算以及和 format() 相同的格式描述符。

为了让你更直观地看到空格是不是被移除掉了，我还利用 f-string 在 newstring 字符串前后增加了“|”，这里可以用任意一个字符，主要用于观察竖线和文字间是否有空格。这样你通过结果，就可以看到“|”和“ 广东省广州市 ”前后的空格已经被自动删除掉了。

可以看到，f-string 和 format() 函数都具有对字符串输出格式调整的功能，不过前者更适用于**字符串和变量连接**的场景，后者更适用于调整字符串的格式。此外，f-string 除了可以用来观察文字对齐的结果外，你还可以在输出变量的同时也输出字符串。

小结

最后，我来为你总结一下这节课的核心内容。今天这节课我们主要讲了四个字符串处理函数，它们也是文本处理工作中最经常用到的函数：

1. format() 函数，可以对字符串或数字进行格式化，对浮点数的小数位数进行调整，实现小数的对齐。
2. split() 函数，按指定的分隔符分隔字段，实现字符串中的部分字符串处理。
3. join() 函数，能够将列表、元组中的每个元素按照指定的分隔符连接成字符串，如果是字符串和变量的连接，你还可以使用 f-sring 方式实现字符串连接功能。
4. strip() 函数，它是自动去掉字符串前后空格的函数，经常用于从文本提取内容后，对文本前后空格进行删除，优化文字内容显示结果的函数。

我把今天这节课用到的代码都放在了 GitHub 上，文稿里也有链接，你可以去学习掌握。

思考题

按照惯例，最后我来为你留一道思考题。如果你需要将文本文件中的“小时: 分钟: 秒”处理成等长格式，再存入 Excel 中，你需要使用哪些函数做怎样的处理，才能实现时间的对齐呢?

欢迎把你的思考和想法写在评论区，我们一起交流讨论。此外，你还可以扫描课程详情页的二维码，加入我们的课程读者群，我也会在群里为你解疑答惑。

提建议

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 11 | 通过程序并行计算，避免CPU资源浪费

下一篇 13 | Excel插件：如何扩展Excel的基本功能?

精选留言 (1)

写留言

Soul of the Drago...

2021-03-09

思考题：先用split函数将“小时”、“分钟”和“秒数”的数字拆开，再分别用format()函数转换格式，再用join()函数将它们重新组合，最后用to_excel()函数存入Excel当中。

作者回复: 点个赞